

# A Fast Apparent-Horizon Finder for 3-Dimensional Cartesian Grids in Numerical Relativity: Supplemental Material

Jonathan Thornburg\*

*Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut,  
Am Mühlenberg 1, D-14476 Golm, Germany*

(Dated: Id: horizon-supplement.tex,v 1.1 2003/10/24 14:30:59 jonathan Exp )

This document contains two appendices which are omitted from the main printed version of this paper in the interests of brevity: appendix B on “Multiprocessor and Parallelization Issues”, and appendix C on “Searching for the Critical Parameter of a 1-Parameter Initial Data Sequence”. These appendices also appear in the preprint-archive version of this paper (gr-qc/0306056).

PACS numbers: 04.25.Dm, 02.70.Bf, 02.60.Cb

Keywords: numerical relativity, apparent horizon, black hole

## APPENDIX B: MULTIPROCESSOR AND PARALLELIZATION ISSUES

CACTUS (like most modern numerical relativity codes using 3-D grids), is designed to run in parallel on multiprocessor computer systems. CACTUS uses a domain-decomposition parallelization scheme, where each processor stores and computes the Einstein equations on its own “chunk” of the spatial grid. Neighboring chunks overlap slightly;<sup>1</sup> CACTUS “synchronizes” them as necessary. An AH may span multiple processors’ grid chunks, and since an AH may move during an evolution, in general we don’t know in advance which processor(s) those are.

Because of the domain decomposition, the multiprocessor “global” interpolator used for the geometry interpolation must in general send each interpolation point to the processor which “owns” that part of the grid, do the interpolation there, and send the results back to the requesting processor. To ensure that every processor has a flow of control in the interpolator code to (potentially) handle interpolation points in its chunk of the grid, the interpolation must be a *collective* operation: code on every processor *must* call the interpolator synchronously (each processor’s code specifying its own choice of interpolation points). Violations of this requirement may result in deadlock in the interprocessor-communication code.

Taking these environmental constraints into account, I have parallelized AHFINDERDIRECT in the following way: To allow the use of standard (uniprocessor) sparse matrix subroutines for solving the Newton’s-method updating routines, AHFINDERDIRECT assigns each AH to a single processor,<sup>2</sup> and searches for that AH only on that processor. However,

---

\*Electronic address: jthorn@aei.mpg.de; URL: <http://www.aei.mpg.de/~jthorn>

<sup>1</sup> This is the 3-D Cartesian grid analog of the angular interpatch ghost zones described in the main printed version of this paper.

<sup>2</sup> A processor may be assigned multiple AHs if there are more AHs than processors.

Part (a): 2 processors				Part (b): 3 or more processors							
processor 1		processor 2		processor 1		processor 2		processor 3		any others	
h/it	what	h/it	what	h/it	what	h/it	what	h/it	what	h/it	what
1/1	$\Theta$	2/1	$\Theta$	1/1	$\Theta$	2/1	$\Theta$	3/1	$\Theta$	—	$\Theta$
1/1	<b>J</b>	2/1	<b>J</b>	1/1	<b>J</b>	2/1	<b>J</b>	3/1	<b>J</b>	—	<b>J</b>
1/2	$\Theta$	2/2	$\Theta$	1/2	$\Theta$	2/2	$\Theta$	3/2	$\Theta$	—	$\Theta$
1/2	<b>J</b>	2/2	<b>J</b>	1/2	<b>J</b>	2/2	<b>J</b>	3/2	<b>J</b>	—	<b>J</b>
1/3	<u><math>\Theta</math></u>	2/3	$\Theta$	1/3	<u><math>\Theta</math></u>	2/3	$\Theta$	3/3	$\Theta$	—	$\Theta$
—	<b>J</b>	2/3	<b>J</b>	—	<b>J</b>	2/3	<b>J</b>	3/3	<b>J</b>	—	<b>J</b>
3/1	$\Theta$	2/4	$\Theta$	—	$\Theta$	2/4	$\Theta$	3/4	<u><math>\Theta</math></u>	—	$\Theta$
3/1	<b>J</b>	2/4	<b>J</b>	—	<b>J</b>	2/4	<b>J</b>	—	<b>J</b>	—	<b>J</b>
3/2	$\Theta$	2/5	<u><math>\Theta</math></u>	—	$\Theta$	2/5	<u><math>\Theta</math></u>	—	$\Theta$	—	$\Theta$
3/2	<b>J</b>	—	<b>J</b>								
3/3	$\Theta$	—	$\Theta$	Abbreviations: h/it = horizon number/iteration number what = what is this processor doing?							
3/3	<b>J</b>	—	<b>J</b>								
3/4	<u><math>\Theta</math></u>	—	$\Theta$								

TABLE I: This table shows two examples of how AHFINDERDIRECT finds multiple horizons in parallel in a multiprocessor environment, for the case where we search for 3 horizons, which are found (the Newton iteration converges, shown by the  $\Theta$ ) after 3, 5, and 4 iterations respectively. The table rows show actions at successive iterations of the algorithm; “—” means a dummy computation (described in the main text). Part (a) shows how the algorithm would work with 2 processors; part (b) shows how it would work with 3 or more processors (the last column refers to any processors other than first 3)

if there are multiple AHs and multiple processors, AHFINDERDIRECT searches for different AHs concurrently on the multiple processors.

All the processors do their Newton iterations synchronously, each processor working sequentially through its own assigned horizon(s), or doing dummy interpolator calls (to preserve the synchronization across all processors) if it has no assigned horizon(s). If/when a processor finishes with a horizon (either locating it or failing to locate it), the processor moves on to its next assigned horizon if there is one, or switches to doing dummy interpolator calls if it has no more assigned horizons left to process. Table I shows two examples of this.<sup>3</sup>

This algorithm requires an explicit global synchronization across all processors at each Newton iteration: After evaluating  $\Theta$ , each processor computes a Boolean flag saying whether that processor needs to continue iterating (this may be true for either or both of two reasons: the Newton iteration hasn’t converged yet on the current horizon, or there is another horizon or horizons assigned to this processor which hasn’t yet been processed). All processors then broadcast their flags, and compute the inclusive-or of all the flags to determine whether to continue the algorithm or exit.

<sup>3</sup> In part (a) of table I, notice that after horizon 1 converges, processor 1 does a dummy **J** computation before starting on the next horizon. This is slightly inefficient, but considerably simplifies the algorithm by keeping the  $\Theta$  and **J** computations synchronized across all processors. In the uniprocessor case this dummy **J** operation is unnecessary, and the algorithm omits it.

```

 $p \leftarrow p_{\text{start}}$ 
 $\delta p \leftarrow \delta p_{\text{start}}$ 
 $G \leftarrow G_{\text{start}}$ 
while ( $|\delta p| \geq \text{tolerance}$ )
{
  try to find a common AH in  $\Sigma[p]$ , using  $G$  as the initial guess
  if (found a common AH)
  then {
     $G \leftarrow \text{the common AH just found}$ 
     $p \leftarrow p + \delta p$ 
  }
  else {
     $\delta p \leftarrow \frac{1}{2}\delta p$ 
     $p \leftarrow p - \delta p$ 
  }
}

```

FIG. 1: This figure shows the continuation-method binary search algorithm for finding the critical parameter  $p$  at which a common AH appears in a 1-parameter family of slices.

### APPENDIX C: SEARCHING FOR THE CRITICAL PARAMETER OF A 1-PARAMETER INITIAL DATA SEQUENCE

In this appendix I describe my continuation-method binary search algorithm for determining the “critical” parameter  $p_*$  at which a common AH appears/disappears in a 1-parameter family of initial data slices,  $p \mapsto \Sigma[p]$ . Without loss of generality I assume that small (large) values of  $p$  do (do not) have a common AH.

The main complication here is that AHFINDERDIRECT needs an initial guess for an AH shape, and if this initial guess is inaccurate AHFINDERDIRECT may fail to find the (an) AH. This means that the obvious binary-search algorithm for finding  $p_*$  isn’t reliable, because a failure to find an AH doesn’t rule out the possible existence of that AH.

Instead, I use a continuation method,  $p$  is “walked up”, using the common AHs found in smaller- $p$  slices as initial guesses for trying to find the common AH in larger- $p$  slices. If the algorithm fails to find a common AH, it decreases  $p$  and tries again with a smaller “walking increment” in  $p$ . Figure 1 shows this algorithm in detail.

# A Fast Apparent-Horizon Finder for 3-Dimensional Cartesian Grids in Numerical Relativity

Jonathan Thornburg\*

*Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut,  
Am Mühlenberg 1, D-14476 Golm, Germany*

In  $3+1$  numerical simulations of dynamic black hole spacetimes, it's useful to be able to find the apparent horizon(s) (AH) in each slice of a time evolution. A number of AH finders are available, but they often take many minutes to run, so they're too slow to be practically usable at each time step. Here I present a new AH finder, `AHFINDERDIRECT`, which is very fast and accurate: at typical resolutions it takes only a few seconds to find an AH to  $\sim 10^{-5}m$  accuracy on a GHz-class processor.

I assume that an AH to be searched for is a Strahlkörper ("star-shaped region") with respect to some local origin, and so parameterize the AH shape by  $r = h(\text{angle})$  for some single-valued function  $h: S^2 \rightarrow \mathbb{R}^+$ . The AH equation then becomes a nonlinear elliptic PDE in  $h$  on  $S^2$ , whose coefficients are algebraic functions of  $g_{ij}$ ,  $K_{ij}$ , and the Cartesian-coordinate spatial derivatives of  $g_{ij}$ . I discretize  $S^2$  using 6 angular patches (one each in the neighborhood of the  $\pm x$ ,  $\pm y$ , and  $\pm z$  axes) to avoid coordinate singularities, and finite difference the AH equation in the angular coordinates using 4th order finite differencing. I solve the resulting system of nonlinear algebraic equations (for  $h$  at the angular grid points) by Newton's method, using a "symbolic differentiation" technique to compute the Jacobian matrix.

`AHFINDERDIRECT` is implemented as a thorn in the `CACTUS` computational toolkit, and is freely available by anonymous CVS checkout.

PACS numbers: 04.25.Dm, 02.70.Bf, 02.60.Cb

Keywords: numerical relativity, apparent horizon, black hole

## I. INTRODUCTION

In  $3+1$  numerical relativity, it's often useful to know the positions and shapes of any black holes in each slice. These are both key physics diagnostics, and potentially valuable for helping choose the coordinate conditions in a numerical evolution. Moreover, black holes inevitably contain singularities, which may need to be excised from the computational domain ([6, 44]).<sup>1</sup> Since the event horizon can be determined only once the entire future development of the slice is known,<sup>2</sup> i.e. only after a numerical evolution is done, in practice one usually uses the apparent horizon(s) as a working approximation which can be computed slice-by-slice while a numerical evolution is still ongoing. (Recall that an apparent horizon

---

\*Electronic address: [jthorn@aei.mpg.de](mailto:jthorn@aei.mpg.de); URL: <http://www.aei.mpg.de/~jthorn>

<sup>1</sup> For more recent work on this topic, see (for example) [2, 3, 11, 15].

<sup>2</sup> For numerical purposes the usual approximate development to a nearly-stationary state suffices ([4, 13, 21, 32]).

is always contained inside an event horizon, and they coincide if the spacetime is stationary ([26]).) Apparent horizons are also interesting due to their close relationship to isolated horizons, which have many useful properties ([7, 22] and references therein).

There has thus been longstanding interest in algorithms and codes to find apparent horizons in numerically computed spacetimes (slices). Here I focus on the case where there are no continuous symmetries such as axisymmetry, and where the spatial grid is Cartesian. Many researchers have developed apparent horizon finding algorithms and codes for this case, for example [5, 10, 25, 27, 28, 30, 37, 41, 42, 45, 51]. However, with the exception of [41, 42],<sup>3</sup> the existing numerical codes for apparent horizon finding are generally very slow, often taking several minutes to find each apparent horizon even on modern computers. This is a serious problem, since we would ideally like to find apparent horizons at each time step of a numerical evolution, and there may be tens of thousands of such time steps.

In this paper I describe a new numerical apparent horizon algorithm and code (based on a generalization of the algorithm and code I described previously for polar-spherical grids ([48])) which is very fast: for typical resolutions it takes only a few seconds to find an AH, so it's practical to run it at every time step of a numerical evolution. This apparent horizon finder is also very accurate, typically finding apparent horizons to within a few tens of parts per million in coordinate position, with similar accuracies for derived quantities such as the apparent horizon area, irreducible mass, coordinate centroid, etc. This apparent horizon finder is implemented as a module ("thorn") AHFINDERDIRECT in the CACTUS computational toolkit ([24], <http://www.cactuscode.org>), and is freely available (GNU GPL licensed) by anonymous CVS checkout.

In the main body of this paper I give a relatively high-level description of AHFINDERDIRECT's algorithms; in the appendices I discuss various technical issues in more detail.

## A. Notation

I generally follow the sign and notation conventions of [53]. In particular, I use the Penrose abstract-index notation, with indices  $i-m$  running over the (Cartesian) spatial coordinates  $x^i \equiv (x, y, z)$  in a (spacelike)  $3+1$  slice.  $g_{ij}$  is the 3-metric in the slice, with associated covariant derivative operator  $\nabla_i$ .  $K_{ij}$  is the extrinsic curvature of the slice (I use the sign convention of [54], not that of [53]) and  $K \equiv K_i^i$  is its trace.  $\eta_{ij}$  is the flat 3-metric. Indices  $uvw$  run over generic angular coordinates  $y^u \equiv (\rho, \sigma)$  on the apparent horizon surface. " $N$ -D" abbreviates " $N$ -dimensional". In cases where the distinction is important, I use a prefix <sup>(3)</sup> to denote quantities defined on a 3-D neighborhood of the apparent horizon surface.

Small-capital indices  $ijk$  label angular grid points on the apparent horizon surface, and  $h[i]$  is the evaluation of a grid function  $h$  at the grid point  $i$ .  $D_u$  and  $D_{uv}$  are finite difference molecules discretely approximating the angular partial derivatives  $\partial_u$  and  $\partial_{uv}$  respectively. If  $m$  is an index into a finite difference molecule  $M$ , then  $M[m]$  is an individual molecule coefficient, and  $m \in M$  means that this coefficient is nonzero. Molecule indices may be obtained by subtracting grid point indices ( $m = j - i$ ), or correspondingly the sum of a grid point index and a molecule index gives a grid point index ( $j = i + m$ ).

---

<sup>3</sup> Schnetter ([41, 42]) has developed an apparent horizon finding algorithm and code quite similar to mine. His work and mine were done independently; neither of us learned of the others' work until our own was mostly complete.

## II. APPARENT HORIZONS

Given a (spacelike)  $3 + 1$  slice, a “marginally trapped surface” (MTS) is defined as a closed spacelike 2-surface in the slice, whose future-pointing outgoing null geodesics have zero expansion  $\Theta$ . In terms of the usual  $3 + 1$  variables this condition becomes ([55])

$$\Theta \equiv \nabla_i n^i + K_{ij} n^i n^j - K = 0 \quad (1)$$

where  $n^i$  is the outward-pointing unit normal to the surface.

An “apparent horizon” (AH) is then defined as an outermost MTS in a slice (there may be multiple MTSs nested inside each other). In this paper I actually describe an algorithm and code for locating MTSs, but since the primary application will be the location of AHs, for convenience of exposition I refer to the MTSs as AHs.

As is common in AH finding, I parameterize the AH surface by first choosing a local coordinate origin  $x_0^i$  inside the AH, then assuming that the horizon is a “Strahlkörper” (“ray body”, or more commonly “star-shaped region”) about this point. A Strahlkörper is defined by Minkowski ([43, p. 108]) as

a region in  $n$ -D Euclidean space containing the origin and whose surface, as seen from the origin, exhibits only one point in any direction.

I take  $y^u \equiv (\rho, \sigma)$  to be generic angular coordinates on the AH surface (or equivalently, on the unit 2-sphere  $S^2$ ). Given these, I then define the AH shape by  $r = h(\rho, \sigma)$ , where  $r \equiv [\sum_i (x^i - x_0^i)^2]^{1/2}$  is a radial coordinate around the local coordinate origin,<sup>4</sup> and the “AH shape function”  $h : S^2 \rightarrow \mathbb{R}^+$  is a single-valued function giving the radius of the AH surface as a function of angular position about the local coordinate origin.

## III. COMPUTING THE EXPANSION $\Theta$ (CONTINUUM)

To write the expansion  $\Theta$  (and thus the AH equation (1)) explicitly in terms of this parameterization, i.e. in terms of  $h$ ’s 1st and 2nd angular derivatives  $\partial_u h$  and  $\partial_{uv} h$ , I first define a scalar function which vanishes on the AH surface and increases outwards,  ${}^{(3)}F \equiv r - h(\rho, \sigma)$ . I then define a (non-unit) outward-pointing normal covector to the AH surface as the gradient of this scalar function,

$$s_i \equiv {}^{(3)}s_i \equiv \nabla_i {}^{(3)}F \quad (2)$$

$$= \partial_i {}^{(3)}F \quad \text{since } F \text{ is a scalar} \quad (3)$$

$$= \partial_i r - \partial_i h \quad (4)$$

$$= \frac{x^i}{r} - X_i^u \partial_u h, \quad (5)$$

where I define the coefficients  $X_i^u \equiv \partial y^u / \partial x^i$ . It’s then straightforward to show that

$$\partial_i s_j = \frac{T_{ij}}{r^3} - X_{ij}^u \frac{\partial h}{\partial y^u} - X_i^u X_j^v \frac{\partial^2 h}{\partial y^u \partial y^v}, \quad (6a)$$

---

<sup>4</sup> Note that I define  $r$  to be the *flat-space* distance from  $x_0^i$  to  $x^i$  – there’s no use of the 3-metric here.

where

$$T_{ij} = \begin{cases} \sum_{k \neq i} (x^k)^2 & \text{if } i = j \\ -x^i x^j & \text{if } i \neq j \end{cases}, \quad (6b)$$

and where I also define the coefficients  $X_{ij}^u \equiv \partial^2 y^u / \partial x^i \partial x^j$ .

The outward-pointing unit normal to the AH surface is then

$$n^i = \frac{s^i}{\|s^k\|} = \frac{g^{ij} s_j}{(g^{k\ell} s_k s_\ell)^{1/2}}, \quad (7)$$

so the expansion  $\Theta$  is given by

$$\Theta \equiv \nabla_i n^i + K_{ij} n^i n^j - K \quad (8)$$

$$= \partial_i n^i + (\partial_i \ln \sqrt{g}) n^i + K_{ij} n^i n^j - K \quad (9)$$

$$= \partial_i \frac{g^{ij} s_j}{(g^{k\ell} s_k s_\ell)^{1/2}} + (\partial_i \ln \sqrt{g}) \frac{g^{ij} s_j}{(g^{k\ell} s_k s_\ell)^{1/2}} + \frac{K^{ij} s_i s_j}{g^{k\ell} s_k s_\ell} - K \quad (10)$$

$$= \frac{A}{D^{3/2}} + \frac{B}{D^{1/2}} + \frac{C}{D} - K, \quad (11)$$

where

$$A = - (g^{ik} s_k)(g^{j\ell} s_\ell) \partial_i s_j - \frac{1}{2} (g^{ij} s_j) \left[ (\partial_i g^{k\ell}) s_k s_\ell \right] \quad (12a)$$

$$B = (\partial_i g^{ij}) s_j + g^{ij} \partial_i s_j + (\partial_i \ln \sqrt{g}) (g^{ij} s_j) \quad (12b)$$

$$C = K^{ij} s_i s_j \quad (12c)$$

$$D = g^{ij} s_i s_j. \quad (12d)$$

Setting  $r = h$  in the definitions (5) and (6) and substituting into (11) and (12) gives  $\Theta$  explicitly in terms of  $h$  and its 1st and 2nd angular derivatives, so the AH equation (1) takes the form

$$\Theta \equiv \Theta(h, \partial_u h, \partial_{uv} h; g_{ij}, K_{ij}, \partial_k g_{ij}) = 0 \quad (13)$$

where the dependence on  $g_{ij}$ ,  $K_{ij}$ , and  $\partial_k g_{ij}$  is implicit through their position dependence (this is discussed in detail in section VI A).

#### IV. SOLVING THE APPARENT HORIZON EQUATION

I view the AH equation (13) as an elliptic PDE for  $h$  on  $S^2$ , and discretize it using standard finite differencing methods: I introduce a total of  $N_{\text{ang}}$  angular grid points  $\{(\rho_i, \sigma_i)\}$  on  $S^2$ , and represent  $h$  and  $\Theta$  by their values  $\{h_i\}$  and  $\{\Theta_i\}$  at these points. Approximating the angular derivatives  $\partial_u h$  and  $\partial_{uv} h$  by finite differencing, (13) then becomes a set of  $N_{\text{ang}}$  nonlinear algebraic equations  $\{\Theta_i = 0\}$  for the  $N_{\text{ang}}$   $\{h_i\}$  values.

I solve these equations by Newton's method in  $N_{\text{ang}}$  dimensions. This in turn has several subparts:

- The actual Newton's-method iteration algorithm
- Computing the (discrete) expansion  $\{\Theta_i\}$  given a (discrete) trial AH shape  $\{h_i\}$

- Computing the Jacobian matrix  $\mathbf{J}_I \equiv d\Theta_I/dh_I$  given a (discrete) trial AH shape  $\{h_I\}$
- Solving the Newton’s-method updating equations  $\mathbf{J} \cdot \delta h = -\Theta$

I describe these in detail in the following sections.

## V. NEWTON’S METHOD

The basic multidimensional Newton’s-method algorithm is well known (see, for example, [47, section 5.3]), but several refinements are necessary for a practical AH finder:

To make Newton’s method converge more robustly if the initial guess is poor, and to limit divergence if the iteration doesn’t converge, AHFINDERDIRECT limits any single Newton step to have an  $\infty$ -norm over the angular grid which is no more than a specified maximum fraction (10% by default) of the mean horizon radius.

Much more sophisticated “modified Newton” algorithms could be used to achieve faster or more robust convergence (eg. [8, 9, 29, 36, 38–40]), but in practice this hasn’t been necessary.<sup>5</sup> In particular, the high-spatial-frequency convergence problems I have previously described for Newton’s-method apparent-horizon-finding ([48]), don’t seem to occur often in practice.

If the slice doesn’t contain an AH (or if either the 3-D Cartesian grid or the  $S^2$  angular grid has insufficient resolution), then the Newton iteration will probably fail to converge. In practice, AHFINDERDIRECT detects this by limiting the Newton iteration to a maximum number of iterations. It’s useful to distinguish between two subcases here:

- If we’re searching for an AH or AHs at each time step of a numerical evolution, and we found this AH at the previous time step, then that AH shape probably provides an excellent initial guess for this step’s Newton iteration, so a relatively low maximum-iterations limit is appropriate. AHFINDERDIRECT uses a default of 10 iterations for this case.
- Otherwise (if we do *not* have a previous-time-step AH as an initial guess), in practice the initial guess is likely to be rather inaccurate, so a higher maximum-iterations limit is appropriate. AHFINDERDIRECT uses a default of 20 iterations for this case.

In addition to the maximum-iterations limit, AHFINDERDIRECT also aborts the finding of an AH if any trial horizon shape  $\{h_I\}$  is outside the 3-D Cartesian grid. Otherwise, AHFINDERDIRECT considers an AH to have been found if and only if the  $\infty$ -norm of the  $\{\Theta_I\}$  values over the angular grid is below a specified threshold ( $10^{-8}$  by default).

For better efficiency, in a multiprocessor environment AHFINDERDIRECT finds multiple AHs in parallel across multiple processors. I describe the algorithm for doing this in appendix B.

---

<sup>5</sup> Additionally, due to the way AHFINDERDIRECT finds multiple AHs in parallel across multiple processors (discussed in detail in appendix B), it would be difficult to use many of the uniprocessor modified-Newton software packages such as [36, 38, 39].



## VI. COMPUTING THE EXPANSION $\Theta$ (DISCRETE)

Given a trial AH shape  $\{h_i\}$ , I compute the expansion  $\{\Theta_i\}$  using (13), approximating the angular derivatives  $\partial_u h$  and  $\partial_{uv} h$  by the usual centered 4th order finite difference molecules  $D_u$  and  $D_{uv}$  respectively. However, there are several complications in this process, which I discuss in the following subsections.

### A. Geometry Interpolation

As shown in section III, the expansion  $\Theta$  implicitly depends on the geometry variables  $g_{ij}$ ,  $K_{ij}$ , and  $\partial_k g_{ij}$  at the AH surface. In practice the geometry variables are only known on a (3-D) Cartesian grid, so they must be interpolated to the AH surface.

Instead of computing the 3-metric derivatives  $\partial_k g_{ij}$  on the full 3-D grid and then interpolating these values to the AH surface, it's much more efficient to do the differentiation only at the AH-surface points, inside the interpolator: An interpolator generally works by (conceptually) locally fitting a fitting function (usually a low-degree polynomial) to the data points in a neighborhood of the interpolation point, then evaluating the fitting function at the interpolation point. By evaluating the *derivative* of the fitting function, the  $\partial_k g_{ij}$  values can be interpolated very cheaply, using only the 3-D input values which are used anyway for interpolating the  $g_{ij}$ .

Even for  $C^\infty$   $g_{ij}$  and  $K_{ij}$ , the usual Lagrange polynomial interpolators give results which are continuous, but *not differentiable* (the interpolated  $\partial_k g_{ij}$  generically has a jump discontinuity) at each position where the interpolator switches the set of input 3-D grid points it uses. (The non-smoothness of interpolation errors is discussed in more detail in [49, Appendix F].) Unfortunately, this lack of smoothness propagates into the AH equation (13), sometimes causing Newton's method to fail to converge. To avoid this problem, I use a (cubic) Hermite interpolator, which guarantees that the interpolated  $g_{ij}$  and  $K_{ij}$  remain differentiable, and that the interpolated  $\partial_k g_{ij}$  remains continuous, even when the interpolator switches input-grid-point sets. Figure 1 shows an example of the smoothness properties of Lagrange and Hermite interpolation, for a simple 1-D model problem.

While the resulting ( $C^0$ ) smoothness of  $\Theta(h)$  isn't quite ideal for Newton's method, in practice it seems to be sufficient not to impair convergence, and attaining a higher degree of smoothness would require a significantly more complicated and expensive interpolator.

### B. Multiple Grid Patches

To avoid  $z$  axis coordinate singularities in the angular computations, I use multiple grid patches to cover  $S^2$ . Figure 2 shows an example of this. In general there are 6 patches, covering neighborhoods of the  $\pm x$ ,  $\pm y$ , and  $\pm z$  axes respectively.

Each patch's nominal grid (shown in thick lines in figure 2) is surrounded by a "ghost zone" (shown in thin lines in figure 2). Once the  $h$  values in the ghost zones are filled in by symmetry operations and/or interpatch interpolation, the finite differencing code can ignore the patch boundaries in computing  $\Theta$ . To keep the interpatch interpolation errors (more precisely, their numerical 2nd derivatives) from dominating those of the 4th order patch-interior angular finite differencing, I use 5th order Lagrange polynomial interpolation. The patch coordinates  $(\rho, \sigma)$  are defined such that adjacent patches always share a common

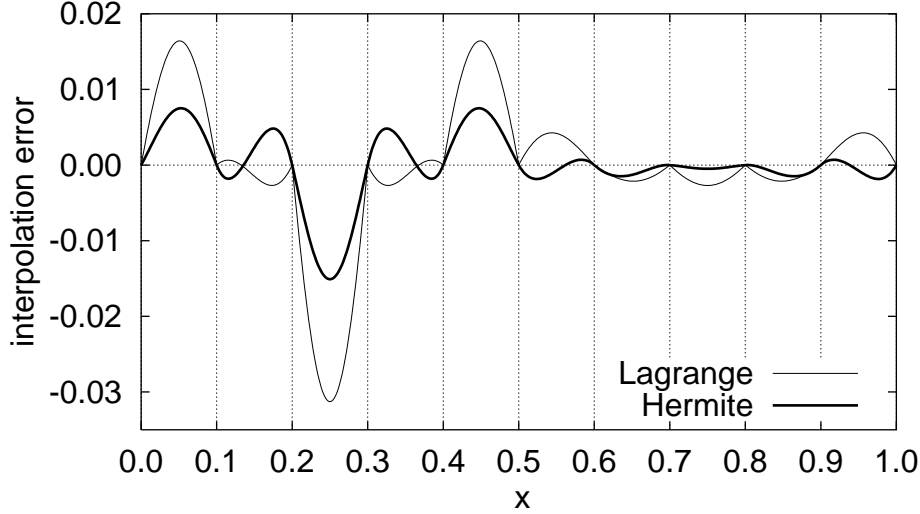


FIG. 1: This figure shows the errors for cubic Lagrange and Hermite interpolation of the function  $f(x) = \exp[\sin(2\pi x)]$  with grid spacing  $\Delta x = 0.1$ . Notice that the Lagrange error (and hence the Lagrange interpolant itself) is non-differentiable at the grid points, whereas the Hermite error (and interpolant) is differentiable everywhere.

angular coordinate, so only 1-D interpolation is required here. I describe the multiple-patch scheme in detail in appendix A.

The Jacobian matrix  $\mathbf{J}_{\text{IJ}}$  must also take into account the ghost-zone symmetry operations and interpatch interpolations. This is conceptually simple, but does require explicitly knowing the Jacobian (i.e. the interpolation coefficients) of the interpatch interpolation. The details are somewhat complicated, and are described in appendix A 3.

The multiple-patch scheme works well, but requires a lot of subtle coding, particularly in handling the ghost-zone updates near patch corners. The overall patch infrastructure is currently about 12K (7K non-blank non-comment) lines of C++ code, out of a total of about 25K (15K) lines of C++ and 2.5K (1.5K) of Maple in AHFINDERDIRECT. In hindsight, a much simpler scheme might well have sufficed to avoid  $z$  axis problems. Notably, [41, 42] reports excellent results using a simple latitude-longitude grid on  $S^2$ , with the grid points staggered across the north/south poles. Another possibility ([23]) would be to have 2 patches meeting at the equator, each using stereographic coordinates.

## VII. COMPUTING THE JACOBIAN MATRIX $\mathbf{J}_{\text{IJ}}$

If there are  $N_{\text{ang}}$  angular grid points, then the Jacobian matrix  $\mathbf{J}_{\text{IJ}} \equiv d\Theta_{\text{I}}/dh_{\text{J}}$  is an  $N_{\text{ang}} \times N_{\text{ang}}$  matrix;  $\mathbf{J}$  is sparse due to the locality of the angular finite differencing. The obvious way to compute  $\mathbf{J}$  is by numerical perturbation: perturb  $h$  at a single angular grid point  $\text{J}$ , then re-evaluate  $\Theta$ <sup>6</sup> and determine the  $\text{J}$ th column of  $\mathbf{J}$  from the changes in  $\Theta$ .

<sup>6</sup> An important optimization is to only re-evaluate  $\Theta$  within an angular-molecule-sized neighborhood of the perturbed point  $\text{J}$ .

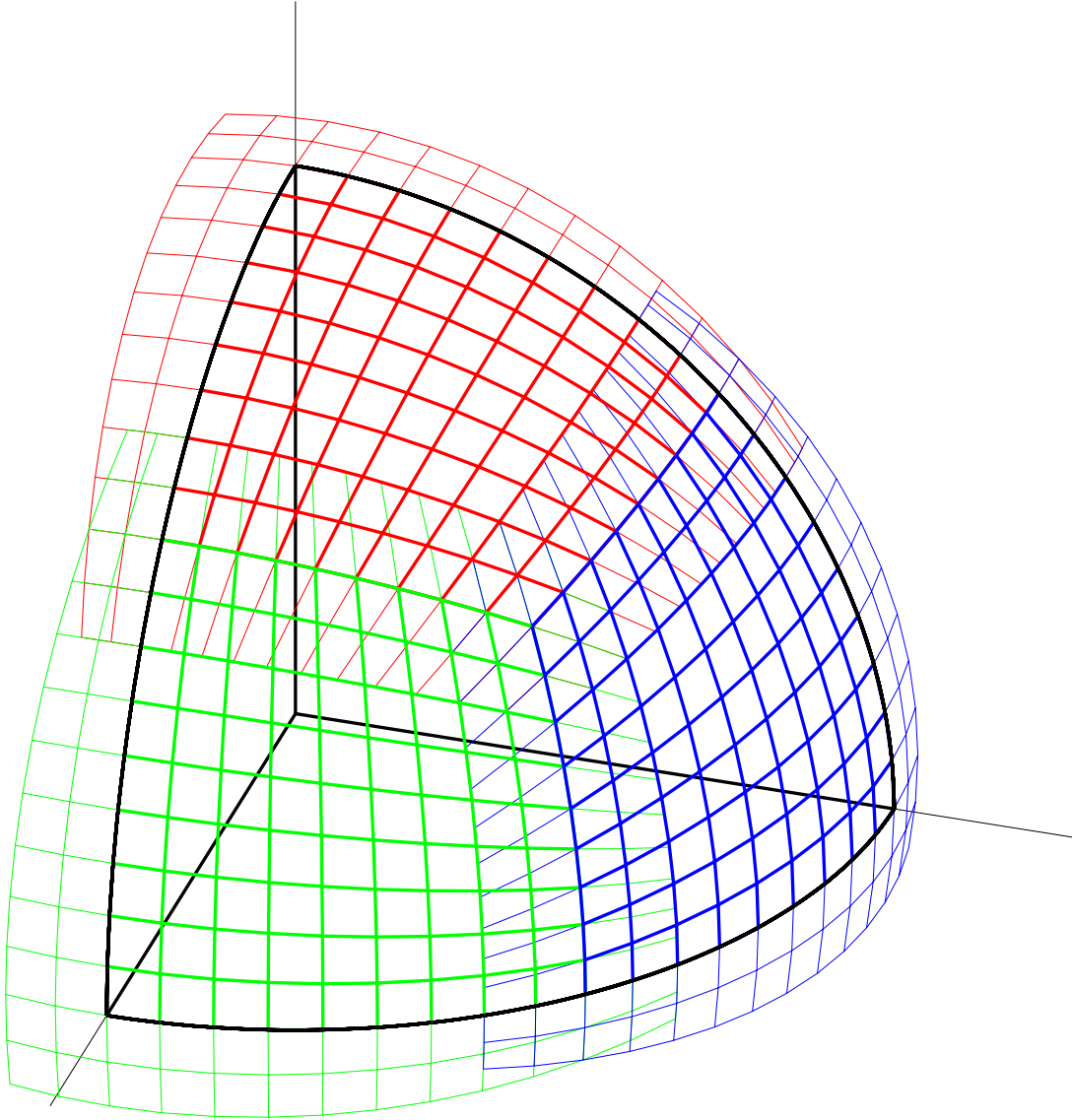


FIG. 2: This figure shows a multiple-grid-patch system covering the  $(+, +, +)$  octant of  $S^2$  with 3 patches, at an angular resolution of  $5^\circ$ . The  $+z$ ,  $+x$ , and  $+y$  patches are shown in red, green, and blue respectively. Each patch’s nominal grid is shown in thick lines; the ghost zones are shown in thin lines.

However, for typical  $N_{\text{ang}}$  values of 300–3000, this is very slow (though its relative simplicity makes it useful for debugging purposes).

Instead of numerical perturbation, `AHFINDERDIRECT` normally uses the “symbolic differentiation” algorithm of [48, section VI] to compute  $\mathbf{J}$  directly from the angular  $\partial_u$  and  $\partial_{uv}$  finite difference molecule coefficients and the (continuum) Jacobian coefficients  $\partial\Theta/\partial(\partial_u h)$  and  $\partial\Theta/\partial(\partial_{uv} h)$ . Temporarily neglecting the interpatch interpolation, the Jacobian is thus

given by

$$\mathbf{J}_{IJ} \equiv \frac{d\Theta_I}{dh_J} = \left\{ \begin{array}{ll} \frac{\partial\Theta}{\partial(\partial_u h)} D_u[J-I] & \text{if } J-I \in D_u \\ 0 & \text{otherwise} \end{array} \right\} + \left\{ \begin{array}{ll} \frac{\partial\Theta}{\partial(\partial_{uv} h)} D_{uv}[J-I] & \text{if } J-I \in D_{uv} \\ 0 & \text{otherwise} \end{array} \right\} + \left\{ \begin{array}{ll} \partial_r \Theta & \text{if } I=J \\ 0 & \text{otherwise} \end{array} \right\} \quad (14)$$

where the first two terms describe the variation in  $\Theta$  at a fixed spatial position with respect to  $h$ , and the last term describes the variation in  $\Theta$  due to a change in  $h$  changing the evaluation position of – and thus the position-dependent coefficients in –  $\Theta$ . Notice that there is no term here for  $\partial\Theta/\partial h$ , since this dependence is included in the  $\partial_r\Theta$  term.

As mentioned in section VIB, the Jacobian (14) must be modified to take into account the ghost-zone symmetry operations and interpatch interpolations. This is described in detail in appendix A 3.

Because  $\Theta$  depends on  $g_{ij}$ ,  $K_{ij}$ , and  $\partial_k g_{ij}$  (cf. (13)), in theory the  $\partial_r\Theta$  term in (14) also requires interpolating  $\partial_k K_{ij}$  and  $\partial_{kl} g_{ij}$  (cf. section VIA). However, doing the computation this way would require a much larger number of interpolations (a total of 80 geometry-interpolator outputs instead of 30), and the expressions for computing  $\partial_r\Theta$  from the interpolated values would be quite complicated.<sup>7</sup>

To avoid these problems, I approximate  $\partial_r\Theta$  by a one-sided radial finite difference,  $\partial_r\Theta \approx [\Theta(h+\varepsilon) - \Theta(h)]/\varepsilon$ , with  $\varepsilon$  typically chosen to be  $10^{-6}$  ([16], [47, pp. 266-267]). Even though this approximation is only  $O(\varepsilon)$  accurate, in practice this doesn't impair the convergence of Newton's method, and it's fairly cheap to compute (one extra  $\Theta(h)$  evaluation per Jacobian computation).

## VIII. SOLVING THE LINEAR SYSTEM $\mathbf{J} \cdot \delta h = -\Theta$

The Jacobian matrix is an  $N_{\text{ang}} \times N_{\text{ang}}$  sparse matrix; for typical angular resolutions  $N_{\text{ang}}$  is in the range 300–3000. Thus for good efficiency it's important to exploit  $\mathbf{J}$ 's sparsity in both storage and computation. I have tried several different linear-equation codes and storage formats: For debugging purposes I have found it very useful to store  $\mathbf{J}$  as a dense matrix and solve the linear system with LAPACK routines.<sup>8,9</sup> For better efficiency I now use either an incomplete-LU-decomposition preconditioned conjugate gradient code ILUCG

<sup>7</sup> The arguments of section VIA would suggest also having the geometry interpolator guarantee at least  $C^0$  continuity of the 2nd derivative values here, although it's not clear if this would actually be necessary in practice.

<sup>8</sup> LAPACK is available from NETLIB (<http://www.netlib.org>).

<sup>9</sup> LAPACK's condition number estimator is a particularly valuable debugging and diagnostic tool. For example, incorrect symmetry boundary conditions often result in  $\mathbf{J}$  being singular (infinite condition number). Another example was in investigating why the Newton iteration sometimes failed to converge in an early version of AHFINDERDIRECT which used Lagrange rather than Hermite interpolation for the geometry variables (cf. section VIA): it was useful to be able to rule out ill-conditioning of the linear system as a possible cause of the convergence failure.

([31]), or the UMFPACK sparse LU-decomposition code ([17–20]);<sup>10,11</sup> both of these codes use the standard “compressed row storage” sparse storage scheme for **J**. Neither code has been entirely satisfactory, so I plan to explore other sparse LU-decomposition codes in the near future.

## IX. PERFORMANCE AND ACCURACY

In this section I outline the general factors affecting AHFINDERDIRECT’s performance (how quickly it can find an AH, or try to find one) and accuracy (how accurately is an AH found). I also briefly compare AHFINDERDIRECT to other AH finders in these respects. I defer detailed numerical results to section X.

### A. Performance

AHFINDERDIRECT’s performance (the time taken to find, or try to find, an AH) depends on two main factors: the total number of angular grid points in the multiple-patch system, and the number of Newton iterations. Since there are no computations done at each Cartesian-grid grid point, the performance is almost independent of the size and resolution of the Cartesian grid.<sup>12</sup>

The total number of angular grid points,  $N_{\text{ang}}$ , is determined by the angular resolution chosen, and whether there are any discrete symmetries in the multiple-patch system. Since practical values of  $N_{\text{ang}}$  vary over roughly an order of magnitude, and empirically the performance scales very roughly as  $N_{\text{ang}}^{1.4}$ , the performance varies over a wide range from this factor alone.

The number of Newton iterations performed by AHFINDERDIRECT is mainly determined by the type of AH being searched for:

- AHFINDERDIRECT is fastest when searching for – and successfully finding – an AH at each time step of a numerical evolution. In this case the AH typically only moves a small distance from one time step to the next, so (using the previous time step’s position as an initial guess for the Newton iteration, cf. section V) typically only 3 Newton iterations are needed to locate it at each time step.
- If AHFINDERDIRECT finds an AH in an initial data slice, typically the initial guess is much less accurate, so 6–10 Newton iterations are needed.
- AHFINDERDIRECT is at its slowest when searching for – but failing to find – an AH at each time step of a numerical evolution. In this case (again cf. section V) it typically takes 20 Newton iterations at each time step.

<sup>10</sup> UMFPACK is available from <http://www.cise.ufl.edu/research/sparse/umfpack>.

<sup>11</sup> UMFPACK also has a condition number estimator, but as of version 4.0 it appears to be unreliable.

<sup>12</sup> On an idealized computer there would be no Cartesian-grid resolution dependence at all, but on actual computers cache effects in the geometry interpolator may cause a slight slowdown at higher Cartesian-grid resolutions.

As discussed in appendix B, AHFINDERDIRECT can search for multiple AHs in parallel on a multiprocessor computer system. In practice, for large-scale runs there are usually (many) more processors available than the number of AHs being searched for. Assuming this, the elapsed time taken to search for all the AHs in parallel is basically the maximum of the time taken to search for each individual AH; this is roughly independent of both the number of AHs searched for, and the number of processors available. Part (b) of the table in appendix B should make this clearer.

## B. Accuracy

The accuracy with which AHFINDERDIRECT can find an AH is mainly determined by the finite differencing errors in the evaluation of the expansion  $\Theta$ . There are two main error contributions: the geometry interpolation from the Cartesian grid to the AH position, and the angular finite differencing within the multiple-patch system on  $S^2$ . (Other error sources such as the interpatch interpolation, the nonzero  $\|\Theta\|$  at which the code considers the Newton iteration to have “converged”, and floating-point roundoff errors, are generally negligible in comparison to the main finite differencing errors.)

For given (smooth)  $g_{ij}$  and  $K_{ij}$ , the errors from the geometry interpolator are determined by the 3-D (Cartesian) grid spacing  $\Delta xyz$ , and by the order of the interpolation scheme. In the limit of small  $\Delta xyz$ , a cubic Hermite geometry interpolator gives  $g_{ij}$  and  $K_{ij}$  to  $O((\Delta xyz)^4)$  and  $\partial_k g_{ij}$  to  $O((\Delta xyz)^3)$ , contributing  $O((\Delta xyz)^3)$  errors to  $\Theta$ . However, at practical resolutions of  $\Delta xyz \sim 0.03m$ – $0.1m$  I find that the convergence is often 0.5–1.0 power of  $\Delta xyz$  better than this, only dropping to the theoretical limits for very high-resolution grids (in practice,  $\Delta xyz \lesssim 0.01m$ ).

AHFINDERDIRECT uses 4th order angular finite differencing within the multiple-patch system on  $S^2$ , which contributes  $O((\Delta\rho\sigma)^4)$  errors to  $\Theta$ , where  $\Delta\rho\sigma$  is the angular resolution.

## C. Comparison to Other AH-Finding Methods

Curvature-flow or fast-flow methods are widely used for AH-finding (see, for example, [25, 30, 45, 51]). Conceptually, a flow method starts with a large 2-surface, and flows this inwards, in such a manner that the flow velocity vanishes on the AH. Unfortunately, this means that the method must move the 2-surface through a large part of the 3-D grid – and thus must do nontrivial computations at a large number of 3-D grid points – before the surface can closely approximate the AH. In contrast, an elliptic-equation method such as that used by AHFINDERDIRECT need only do computations on a 2-D set of (AH-surface) grid points, so it can potentially be much faster.

However, a flow method can (at least modulo numerical errors) guarantee to find the *outermost* MTS in a slice, whereas an elliptic-equation method is only locally convergent, and hence offers no information on what other MTSs might be outside any “AH” it finds.

Another common class of AH-finding methods are function-minimization methods such as those described by [5, 10]. These parameterize a trial AH surface by spherical harmonic or other spectral coefficients, define a surface-integral error norm  $\int \Theta^2 dA$  which has a global minimum of 0 at the AH surface, then use a general-purpose function-minimization algorithm to minimize the error norm over the surface-coefficient space. These methods are inherently quite slow because (for a generic slice with no continuous symmetries) they must

determine a fairly large number of surface coefficients, and the generic function-minimization algorithm only “learns” a single number (the error norm) for each surface evaluation, and thus requires many surface evaluations to converge. For example, using a spherical harmonic expansion up to order  $N$  to parameterize the AH surface, there are  $O(N^2)$  surface coefficients, so  $O(N^2)$  iterations are needed to converge. Each iteration takes  $O(N^2)$  work to evaluate the surface integral, so the total work is  $O(N^4)$ . The exponential convergence of spectral series allows  $N$  to be chosen to be fairly small for a given surface accuracy, but in practice function-minimization AH finders are still very slow.

Minimization methods are also inherently somewhat limited in their accuracy, because the location of the error norm’s minimum is very sensitive to small numerical errors. (In general, relative errors of  $O(\varepsilon)$  in a smooth function result in relative errors of  $O(\sqrt{\varepsilon})$  in the location of the function’s minima.)

#### D. What makes AHFINDERDIRECT Fast?

Based on the above analyses, I think the key algorithm component which makes AHFINDERDIRECT fast is the posing of the AH equation (1) as an elliptic PDE on  $S^2$  for the AH shape function  $h$ . Given this, I believe that any efficient implementation would result in an AH finder with roughly the same performance and accuracy as AHFINDERDIRECT.

A notable example of this is Schnetter’s AH finder ([41, 42]), which poses the AH equation in the manner as mine, but uses a rather different finite differencing scheme and solution method for the finite difference equations. We have not yet made a detailed comparison of our AH finders, but it appears they are broadly comparable in performance and accuracy.

Huq’s AH finder ([27, 28]) also poses the AH equation as an elliptic PDE on  $S^2$ , but he uses Cartesian-grid finite differencing to evaluate the surface expansion and Jacobian matrix, rather than the angular-grid finite differencing which Schnetter and I use. Because of this, and because he uses numerical perturbations to compute the Jacobian matrix (cf. section VII), Huq’s AH finder is roughly an order of magnitude slower than mine.

### X. SAMPLE RESULTS

In this section I present various sample results to test and demonstrate AHFINDERDIRECT’s performance. For comparison, I also show some results for another AH finder implemented in the CACTUS toolkit, the fast-flow method of [25].<sup>13</sup> (This was the main CACTUS AH finder prior to AHFINDERDIRECT.) Although some of the test slices are in fact axisymmetric, I configured both AH finders to treat the slices as fully 3-D, with only the discrete symmetries of reflection across the  $x$ ,  $y$ , and/or  $z = 0$  planes as appropriate. All timings are user-mode CPU times on a 1.7 GHz dual Pentium IV processor system (256 KB cache per processor) with 1.0 GB of memory.

---

<sup>13</sup> Cactus thorn AHFINDER, slightly modified to allow a spherical harmonic expansion up to degree  $\ell_{\max} = 50$  (by default the limit is 19). (As discussed below, in practice AHFINDER is limited to  $\ell_{\max} \lesssim 20$ .)

### A. Boosted Kerr Slices

As a first test case, I first consider Kerr spacetime in Kerr-Schild coordinates ([35, exercise 33.8]), where the AH is a coordinate ellipsoid with radii (semi-major axes)

$$r_z = (1 + \sqrt{1 - a^2})m \quad (15a)$$

$$r_x = r_y = r_z \sqrt{1 + \left(\frac{am}{r_z}\right)^2} = \sqrt{\frac{2r_z}{m}} m \quad (15b)$$

and area

$$A = 4\pi(r_z^2 + a^2m^2) \quad (15c)$$

where  $a = J/m^2$  is the black hole’s dimensionless angular momentum. I then Lorentz-boost this with a velocity  $v$  in the  $x$  direction. The horizon area is invariant under the boost, but in the code’s coordinate system, length-contraction makes the AH a triaxial ellipsoid, and the interaction of the black hole’s spin and the boost results in the slice *not* being symmetric across either the  $x = 0$  or the  $y = 0$  plane.

Table I shows the accuracy and performance of AHFINDERDIRECT and the fast flow AH finder on various boosted-Kerr slices, for a number of choices of the various numerical parameters.

The first section of the table shows AHFINDERDIRECT’s behavior as the resolution of the underlying Cartesian grid is varied, using the default cubic Hermite geometry interpolator. At very low resolution ( $\Delta xyz = 0.2$ ) AHFINDERDIRECT fails to find the AH, due to the geometry interpolation “seeing” the Kerr ring singularity. At higher resolution (decreasing  $\Delta xyz$ ) the accuracy improves rapidly, until it levels out at high resolutions due to the angular finite differencing errors. For the computer system used here, the time taken to find the AH is essentially independent of the Cartesian grid resolution.

The second section of the table shows AHFINDERDIRECT’s behavior as the resolution of the underlying Cartesian grid is varied, using a lower-order (quadratic) geometry interpolator. Compared to the default (cubic) geometry interpolator, this makes AHFINDERDIRECT a factor of 2 to 3 faster, and roughly an order of magnitude less accurate. Also, at the very lowest resolution AHFINDERDIRECT is now able to find the AH, when it couldn’t find it using the cubic interpolator.

Comparing the first two sections of the table shows that changing the interpolation order seems to make only a minor difference to the fast flow method’s behavior; all the remaining tests use its default (quadratic Lagrange) geometry interpolator. As discussed in section IX C, the fast flow method becomes much slower at high Cartesian grid resolutions.

The third section of the table shows AHFINDERDIRECT’s behavior as the angular resolution is varied. As the resolution is increased (decreasing  $\Delta\rho\sigma$ ) AHFINDERDIRECT becomes slower but more accurate, until the error levels off at high angular resolutions due to the Cartesian-grid geometry interpolation errors.

The third section of the table also shows the fast flow method becoming slower as its resolution parameter  $\ell_{\max}$  is increased. Unfortunately, beyond  $\ell_{\max} \approx 10$  the method’s accuracy stops improving and begins to worsen, and beyond  $\ell_{\max} \approx 20$  the fast flow method fails to find the AH. I suspect this is due to numerical ill-conditioning, but I haven’t investigated this in detail.

The fourth section of the table shows AHFINDERDIRECT’s behavior when the local coordinate origin is offset from the coordinate origin. Notice that the accuracy with which



						AHFinderDirect					Fast Flow				
$a$	$v_x$	$(r_x/\gamma, r_y, r_z)$	$\Delta xyz$	origin	notes	$\Delta\rho\sigma$	interp	$N_{\text{ang}}$	time	$\ \delta h\ _{\text{rms}}$	$(\delta A)/A$	$\ell_{\text{max}}$	interp	time	$(\delta A)/A$
0.8	0.8	(1.07, 1.79, 1.60)	0.20	( 0.0, 0.0)		5°	H3	1121	2.0	failed	failed	10	L3	25	$1.2 \times 10^{-2}$
0.8	0.8	(1.07, 1.79, 1.60)	0.15	( 0.0, 0.0)		5°	H3	1121	4.0	$3.4 \times 10^{-4}$	$4.7 \times 10^{-4}$	10	L3	26	$6.5 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.10	( 0.0, 0.0)		5°	H3	1121	4.1	$5.4 \times 10^{-5}$	$7.8 \times 10^{-5}$	10	L3	33	$2.2 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		5°	H3	1121	4.2	$2.5 \times 10^{-5}$	$4.1 \times 10^{-5}$	10	L3	96	$4.6 \times 10^{-4}$
0.8	0.8	(1.07, 1.79, 1.60)	0.03	( 0.0, 0.0)		5°	H3	1121	4.3	$2.6 \times 10^{-5}$	$4.1 \times 10^{-5}$	10	L3	350	$1.0 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.20	( 0.0, 0.0)		5°	H2	1121	1.8	$3.6 \times 10^{-3}$	$4.7 \times 10^{-3}$	10	L2	24	$1.1 \times 10^{-2}$
0.8	0.8	(1.07, 1.79, 1.60)	0.15	( 0.0, 0.0)		5°	H2	1121	1.8	$4.0 \times 10^{-4}$	$4.3 \times 10^{-4}$	10	L2	26	$6.2 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.10	( 0.0, 0.0)		5°	H2	1121	1.9	$6.5 \times 10^{-4}$	$1.0 \times 10^{-3}$	10	L2	32	$2.1 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		5°	H2	1121	1.7	$1.3 \times 10^{-4}$	$2.4 \times 10^{-4}$	10	L2	95	$4.5 \times 10^{-4}$
0.8	0.8	(1.07, 1.79, 1.60)	0.03	( 0.0, 0.0)		5°	H2	1121	2.0	$3.4 \times 10^{-5}$	$4.7 \times 10^{-5}$	10	L2	350	$1.0 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		7.5°	H3	533	2.0	$1.3 \times 10^{-4}$	$2.0 \times 10^{-4}$	7	L2	69	$2.7 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		5.0°	H3	1121	4.2	$2.5 \times 10^{-5}$	$4.1 \times 10^{-5}$	10	L2	95	$4.5 \times 10^{-4}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		3.0°	H3	2945	13	$4.4 \times 10^{-6}$	$7.3 \times 10^{-6}$	15	L2	170	$6.9 \times 10^{-4}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		1.8°	H3	7905	43	$1.3 \times 10^{-6}$	$1.8 \times 10^{-6}$	20	L2	280	$1.3 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		1.0°	H3	25025	220	$9.5 \times 10^{-7}$	$1.3 \times 10^{-6}$	28	L2	2600	failed
0.8	0.8	(1.07, 1.79, 1.60)	0.05	(−0.5, −0.9)	(a)	5°	H3	1121	5.7	$2.4 \times 10^{-5}$	$3.3 \times 10^{-5}$	10	L2	960	$1.2 \times 10^{-3}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	( 0.0, 0.0)		5°	H3	1121	4.2	$2.5 \times 10^{-5}$	$4.1 \times 10^{-5}$	10	L2	95	$4.5 \times 10^{-4}$
0.8	0.8	(1.07, 1.79, 1.60)	0.05	(+0.5, +0.9)	(a)	5°	H3	1121	7.2	$1.7 \times 10^{-5}$	$2.7 \times 10^{-6}$	10	L2	960	$5.6 \times 10^{-3}$
0.99	0.8	(0.91, 1.51, 1.14)	0.03	( 0.0, 0.0)		3°	H3	2945	15	$3.3 \times 10^{-6}$	$4.8 \times 10^{-6}$	15	L2	2300	failed
0.999	0.8	(0.87, 1.45, 1.04)	0.03	( 0.0, 0.0)		3°	H3	2945	16	$9.5 \times 10^{-6}$	$1.4 \times 10^{-5}$	15	L2	1600	failed
0.999	0.95	(0.45, 1.45, 1.04)	0.02	( 0.0, 0.0)	(b,d)	3°	H3	2945	13	$4.0 \times 10^{-4}$	$6.5 \times 10^{-4}$	not tested on this slice			
0.999	0.98	(0.29, 1.45, 1.04)	0.02	( 0.0, 0.0)	(c,d)	3°	H3	2945	12	$1.3 \times 10^{-4}$	$1.1 \times 10^{-4}$	not tested on this slice			

Table columns not described in the main text:

origin The  $(x, y)$  components of the local coordinate origin (the  $z$  component is always 0)

interp The geometry interpolator:

H(L) means Hermite (Lagrange) polynomial interpolation,

the following integer gives the order

time user-mode CPU time in seconds

$\|\delta h\|_{\text{rms}}$  The rms-norm over the angular grid of the error in the computed AH radius  $h$

$(\delta A)/A$  The relative error in the computed AH area

$\ell_{\text{max}}$  The maximum order of the spherical harmonic expansion

<sup>a</sup>Fast-flow initial guess changed to a coordinate sphere of radius 2.5, and Cartesian grid enlarged to size  $\pm 4$  (the larger Cartesian grid size points should have only minimal effects on AHFINDERDIRECT's performance, but should slow the fast flow method by a factor of  $(4/2.5)^3 \approx 4$ )

<sup>b</sup>AHFINDERDIRECT initial guess changed to a coordinate ellipsoid of radii (0.5, 1.5, 1.0)

<sup>c</sup>AHFINDERDIRECT initial guess changed to a coordinate ellipsoid of radii (0.3, 1.5, 1.0)

<sup>d</sup>Cartesian grid shrunk to size  $\pm 2$  to reduce memory usage (this should have only minimal effects on AHFINDERDIRECT's performance)

TABLE I: This table shows the accuracy and performance of AHFINDERDIRECT on various boosted Kerr slices. In each case the black hole has dimensionless rest mass  $m = 1$ . Except as noted, the Cartesian grid is of size  $\pm 2.5$  (more precisely,  $[-2.5, +2.5]$  in  $x$  and  $y$  and  $[0, 2.5]$  in  $z$ , with  $z \leftrightarrow -z$  reflection symmetry across the  $z = 0$  plane). Except as noted, the AHFINDERDIRECT initial guess is a coordinate sphere of radius 1.5, and the fast-flow initial guess is a coordinate sphere of radius 2. AHFINDERDIRECT used the ILUCG sparse matrix routines in all cases. In most cases the  $\infty$ -norm error in the AHFINDERDIRECT AH shape was less than twice the rms-norm error shown here; in no case did it exceed 5 times the rms-norm error.

the AH is found isn't significantly changed, and the time taken to find the AH is only mildly increased, even when the local coordinate origin is offset by up to  $\frac{1}{2}$  the AH radius. The fast flow method is still able to find the AH with the offset local coordinate origins, but it requires changes to the initial guess, and (even after correcting for the larger grid) it slows dramatically and becomes less accurate.

The final section of the table shows AHFINDERDIRECT's behavior on some more difficult boosted-Kerr slices, where the spin is closer to maximal and/or the boost is larger. Because the ring singularity in Kerr moves closer to the AH at high spins, and length contraction makes the AH strongly triaxial at high boosts, these tests used higher Cartesian and angular resolutions than the previous tests. AHFINDERDIRECT still finds the horizon rapidly and with high accuracy in these cases, although in the two most difficult cases quite good initial guesses were required. The fast flow method is not able to find the AH for any of these cases, even with some adjustment of its initial guesses (this may be due in part to its user interface only allowing for axisymmetric initial guesses).

Across all the boosted-Kerr tests, AHFINDERDIRECT is roughly an order of magnitude faster, and two orders of magnitude more accurate, than the fast flow method.

## B. Misner and Brill-Lindquist Slices

The Misner ([33, 34]) and Brill-Lindquist ([12]) initial data slices are standard test problems in numerical relativity. Both are time-symmetric ( $K_{ij} = 0$ ), 3-conformally-flat ( $g_{ij} = \Psi\eta_{ij}$  for some spatially-varying conformal factor  $\Psi$ ), and (for suitable values of their parameters) may contain any number  $N \geq 1$  of black holes.

The simplest case of Misner data (and the only case I consider here) is that of 2 throats, each of bare mass unity. Here the conformal factor is

$$\Psi = 1 + \sum_{n=1}^{\infty} \frac{1}{\sinh(n\mu)} \left( \frac{1}{r_n^+} + \frac{1}{r_n^-} \right) \quad (16)$$

where

$$r_n^{\pm} = \sqrt{x^2 + y^2 + [z \pm \coth(n\mu)]^2} \quad . \quad (17)$$

with  $\mu > 0$  is a real parameter. The individual throats are located at coordinate positions  $(0, 0, \pm \coth \mu)$ . For small  $\mu$  there is only a single AH enclosing both throats, while for large  $\mu$  there are individual AHs enclosing each throat, but no common AH enclosing both throats.

The conformal factor for  $N$ -throat Brill-Lindquist initial data is

$$\Psi(\vec{x}) = 1 + \frac{1}{2} \sum_{i=1}^N \frac{m_i}{|\vec{x} - \vec{x}_i|} \quad (18)$$

where the  $i$ th throat has bare mass  $m_i$  and is located at the coordinate position  $\vec{x}_i$ . Here I consider the cases  $N = 2$  and  $N = 3$ , where the throats each have bare mass unity, and are uniformly spaced in a coordinate circle of radius  $R > 0$ . Similarly to the Misner data, for small  $R$  there is a single common AH, while for large  $R$  there are  $N$  individual AHs but no common AH.

The AH-finder test problem I consider here is to numerically determine the “critical” value of the parameter ( $\mu$  for Misner,  $R$  for Brill-Lindquist) at which the common horizon

Test Problem	Alcubierre <i>et al.</i>		AHFINDERDIRECT	
	parameter	critical parameter	critical parameter	critical AH area
Misner	$\mu$	1.364	$1.365\,071\,172 \pm 3$	$409.549\,358 \pm 3$
Brill-Lindquist 2-throat	$R$	0.766	$0.766\,197\,45 \pm 5$	$196.407\,951 \pm 3$
Brill-Lindquist 3-throat	$R$	$1.18 \pm 4$	$1.195\,499\,53 \pm 5$	$444.756\,224 \pm 3$

TABLE II: This table shows the maximum Misner  $\mu$  and Brill-Lindquist  $R$  for which AHFINDERDIRECT found a common AH, along with the area of that common AH. All uncertainties are in units of the last digits shown. For the 2-throat Brill-Lindquist data, other values in the literature include  $R = 0.767 \pm 1$  ([52]) and  $R = 0.768$  ([45]). However, [45] report a critical AH area for this case of 184.16, about 6% different from AHFINDERDIRECT’s. I don’t know the cause of this discrepancy.

appears/disappears for each family of slices. To do this, I used the CACTUS thorn IDANALYTICBH to construct the initial data slices, approximating the infinite sum (16) by its first 30 terms.<sup>14</sup> For each of a number of combinations of the CACTUS Cartesian grid spacing and the AHFINDERDIRECT angular grid spacing,<sup>15</sup> I used a continuation-method binary search (described in detail in appendix C) to determine the critical parameter. I did convergence tests ([14]) in both grid spacings to verify that the values shown are reliable estimates of the true continuum values, and I used Richardson extrapolation in the angular grid spacing to improve the accuracy.<sup>16</sup> Table II shows the results, together with values reported by [1] for comparison. The AHFINDERDIRECT values are in excellent agreement with those of [1], and are dramatically more accurate.

### C. Binary Black Hole Collision Spacetimes

As a final example, I consider the binary black hole collision evolution described in table III. Figure 3 shows the AH areas found by AHFINDERDIRECT and the fast flow AH finder for this evolution. For the AHs they both find, the two AH finders agree very well. AHFINDERDIRECT found the outer common AH somewhat sooner than the fast flow AH finder ( $t = 4.633$  ( $3.64m$ ) versus  $t = 5.50$  ( $4.32m$ )), and was the only finder to find the inner common AH. Figure 4 shows the 3 AHs found by AHFINDERDIRECT at two times during the evolution.

For this evolution the mean CPU times per time step were 5.2 seconds for AHFINDERDIRECT and (for those time steps for which it ran) 55 seconds for the fast flow AH finder,

<sup>14</sup> Raising this to 50 terms changed the numerically computed critical  $\mu$  by  $< 10^{-12}$ , and the horizon area by  $< 10^{-7}$ .

<sup>15</sup> I used very high resolutions here, with grid spacings as small as 0.01 for the CACTUS 3-D grid and  $0.5^\circ$  for the AHFINDERDIRECT angular grid.

<sup>16</sup> Because the AHFINDERDIRECT angular grid isn’t commensurate with the CACTUS Cartesian grid, the geometry-interpolation errors effectively have a quasirandom phase at each angular grid point. This prevents these errors from being smooth enough to allow Richardson extrapolation on the Cartesian grid spacing. However, the variation of the computed critical parameters with the Cartesian grid spacing can still be used qualitatively to help estimate the critical parameters’ accuracy.

---

Initial data	Misner $\mu = 2.0$ (ADM mass $m = 1.272$ ) black holes located on the $z$ axis at $z = \pm 0.99$
Coordinates	$1 + \log$ lapse, $\Gamma$ -driver shift
Numerical grid	$\Delta xyz = 0.0666$ ( $0.052m$ ); $+xyz$ octant symmetry
Time integration	iterated Crank-Nicholson (3 iterations) Courant number $\Delta t / \Delta xyz = 0.25$
Outer boundaries	in the computational coordinates the outer boundaries were at $xy_{\max} = 5.37$ ( $4.22m$ ), $z_{\max} = 6.43$ ( $5.05m$ ); a “fisheye” nonuniform-grid transformation was used which placed the physical outer boundaries at $xy_{\max} = 13.1$ ( $10.3m$ ), $z_{\max} = 16.3$ ( $12.8m$ )
AHFINDERDIRECT	searching for individual and inner/outer common AHs at each time step cubic Hermite geometry interpolation angular resolution $\Delta\rho\sigma = 5^\circ$ for individual AH ( $N_{\text{ang}} = 580$ , $+xy$ quadrant symmetry, local coordinate origin $(0, 0, 0.85)$ ) angular resolution $\Delta\rho\sigma = 3^\circ$ for inner and outer common AHs ( $N_{\text{ang}} = 768$ , $+xyz$ octant symmetry, local coordinate origin $(0, 0, 0)$ ) UMFPACK sparse matrix routines
Fast flow AH finder	searching for individual and common AHs each 10 time steps fast flow method ([25]), spherical harmonics up to $\ell_{\max} = 10$ same local coordinate origins as AHFINDERDIRECT

---

TABLE III: This table gives various parameters for the binary black hole collision evolution shown in figures 3 and 4.

so despite searching for 3 AHs instead of 2, AHFINDERDIRECT was about an order of magnitude faster than the fast flow method.

Since the runs just described, I have changed AHFINDERDIRECT’s default geometry interpolator from cubic to quadratic Hermite. In practice AHFINDERDIRECT is usually used in numerically computed slices whose geometries have numerical errors large enough to dominate AHFINDERDIRECT’s intrinsic errors. Thus (cf. section X A) the lower-order geometry interpolation makes little difference to the practical accuracy with which the apparent horizons are found, and it speeds up AHFINDERDIRECT by roughly a factor of 2 to 3. For example, in a recent large binary-black-hole collision simulation (details of which will be reported elsewhere), AHFINDERDIRECT (using the UMFPACK sparse matrix routines) averaged 1.7 seconds per time step, as compared with 61 seconds per time step for the fast flow AH finder.

## XI. CONCLUSIONS

In this paper I present a detailed description of a new numerical apparent horizon finder for 3-dimensional Cartesian grids, AHFINDERDIRECT. AHFINDERDIRECT is typically at least an order of magnitude faster than other widely-used apparent horizon finders; in particular AHFINDERDIRECT is fast enough that it’s practical to find apparent horizons at each time step of a numerical evolution. This allows apparent horizon positions to readily

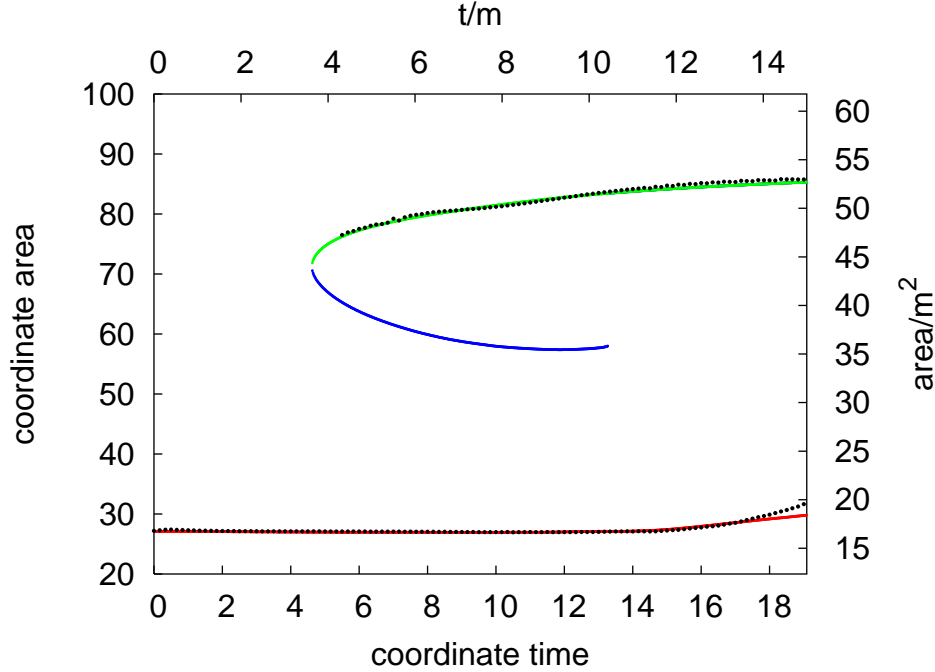


FIG. 3: This figure shows the areas of the various AHs in the Misner  $\mu = 2.0$  collision described in table III. The black points are the areas found by the fast flow AH finder; the other curves are all from AHFINDERDIRECT. The gradual rise in the area of the outer common AH after  $t \approx 9$ , and in the area of the individual AH after  $t \approx 15$ , is due to outer boundary reflections making the overall evolution inaccurate.

be used in coordinate conditions (see, for example, [46]) or for other diagnostic purposes.

AHFINDERDIRECT is also very accurate, typically finding apparent horizons to within  $\sim 10^{-5}m$  in coordinate position.

AHFINDERDIRECT is implemented within the CACTUS computational toolkit, and is freely available (GNU GPL licensed) by anonymous CVS checkout from `cvs.aei.mpg.de:/numrelcvs` in the directory `AEIThorns/AHFinderDirect`. It would also be fairly easy to port AHFINDERDIRECT to a different (non-CACTUS) numerical relativity code.

## APPENDIX A: DETAILS OF THE MULTIPLE PATCH SYSTEM

### 1. Coordinates

I define angular coordinates on  $S^2$  based on rotation angles about the local  $xyz$  coordinate axes:

$$\begin{aligned}
 \mu &= \text{rotation angle about the local } x \text{ axis} = \arctan(y/z) \\
 \nu &= \text{rotation angle about the local } y \text{ axis} = \arctan(x/z) \\
 \phi &= \text{rotation angle about the local } z \text{ axis} = \arctan(y/x)
 \end{aligned}
 \tag{A1}$$

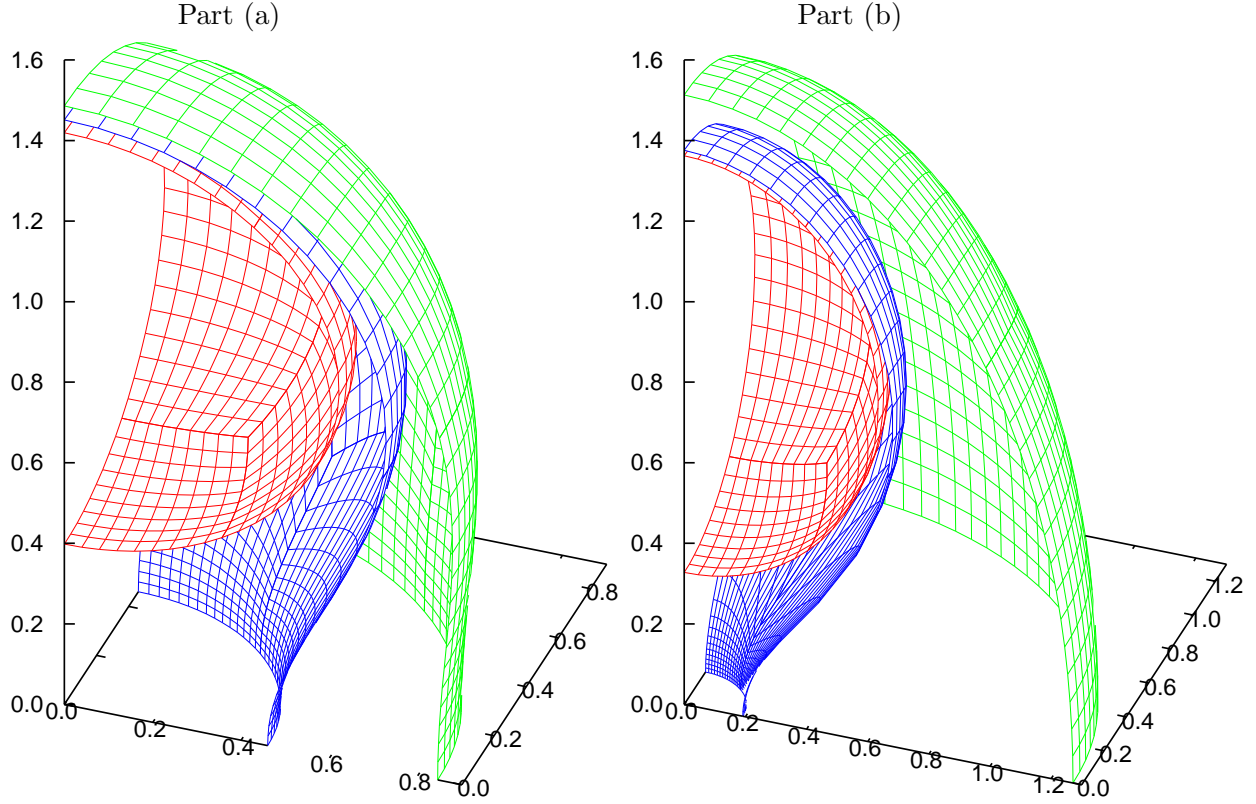


FIG. 4: This figure shows the three AHs in the Misner  $\mu = 2.0$  collision described in table III. Part (a) shows the horizons at  $t = 5.00$  ( $3.93m$ ); part (b) shows them at  $t = 8.00$  ( $6.28m$ ). In both parts the color coding matches that of figure 3.

where all the arctangents are 4-quadrant based on the signs of  $x$ ,  $y$ , and  $z$ . I then define coordinate patches covering neighborhoods of the  $\pm z$ ,  $\pm x$ , and  $\pm y$  axes, using the generic patch coordinates

$$\begin{aligned}
 \pm z \text{ patch has generic patch coordinates } (\rho, \sigma) &= (\mu, \nu) \\
 \pm x \text{ patch has generic patch coordinates } (\rho, \sigma) &= (\nu, \phi) \\
 \pm y \text{ patch has generic patch coordinates } (\rho, \sigma) &= (\mu, \phi)
 \end{aligned}
 \tag{A2}$$

The resulting set of 6 patches cover  $S^2$  without coordinate singularities.<sup>17</sup> Alternatively, if the slice has  $z \leftrightarrow -z$  reflection symmetry about the local coordinate origin, then the 5 patches  $+z$ ,  $\pm x$ , and  $\pm y$  cover the  $+z$  hemisphere of  $S^2$ . Similarly, suitable sets of 4 or 3 patches may be used to cover quadrants or octants of  $S^2$  respectively; figure 2 shows an example of this last case.

<sup>17</sup> Another way to visualize these patches and coordinates is to imagine an  $xyz$  cube with  $xyz$  grid lines painted on its face. Now imagine the cube to be flexible, and inflate it like a balloon, so it becomes spherical in shape. The resulting coordinate lines will closely resemble those for  $(\mu, \nu, \phi)$  coordinates.

## 2. Ghost Zones

Each patch is a rectangle in its own  $(\rho, \sigma)$  coordinates; I use the usual “ghost zone” technique for handling finite differencing near the patch boundaries. I refer to the non-ghost-zone part of a patch’s grid as its “nominal” grid. Adjacent patches’ nominal grids just touch. (Grid-function values in) the ghost zones are filled in from values in their own and other patches’ nominal grids by symmetry operations and/or interpatch interpolations.

With the coordinate choice (A2), adjacent patches always share the angular coordinate perpendicular to their mutual boundary, so the interpatch interpolations need only be done in one dimension, in the direction parallel to the boundary. Since off-centering an interpolant, particularly a high-order one, significantly degrades its accuracy, I have tried to design the algorithms to keep the interpolations centered wherever possible.<sup>18</sup>

The most complicated part of the multiple-patch scheme is in the handling of the “corner” ghost-zone grid points, those ghost-zone grid points which are outside their patch’s nominal grid in both the  $\rho$  and the  $\sigma$  directions. Figure 5 shows the three basic cases:

- (a) Figure 5(a) shows an example of a corner between two symmetry ghost zones. In this case it takes 2 sequential symmetry operations (shown by the curved arrows) to fill in the corner from the nominal grid. Fortunately, symmetry operations commute, i.e. the results are independent of the order of the two symmetry operations.
- (b) Figure 5(b) shows an example of a corner between a symmetry and an interpatch ghost zone. To keep the interpolations centered, I use a 3-phase algorithm here:
  1. Use symmetry operations (for example, the one shown by the dotted arrow in the figure) to fill in the non-corner ghost-zone points in the neighboring patch.
  2. Do a centered interpatch interpolation from the neighboring patch to this patch; this interpolation may use some points from the neighboring patch’s ghost zone.
  3. Use symmetry operations (for example, the one shown by the solid arrow in the figure) to fill in the corner ghost-zone points in this patch.
- (c) Figure 5(c) shows an example of a corner between two interpatch ghost zones (this only happens when 3 patches meet at a corner). This case requires only a single interpatch interpolation for each ghost-zone grid point.

AHFINDERDIRECT actually uses the following 3-phase algorithm (which includes each of (a)–(c) above as special cases) to perform all the necessary symmetry operations and interpatch interpolations across all patches, in a correct order:<sup>19</sup>

1. Use symmetry operations to fill in the non-corner parts of all symmetry ghost zones in all patches.
2. Use interpatch interpolations to fill in all interpatch ghost zones in all patches.

<sup>18</sup> Another reason to keep the interpolations centered in AHFINDERDIRECT was to allow re-use of the multiple-patch software from an earlier time evolution code ([50]), where centering the interpolations helped keep the evolution stable.

<sup>19</sup> The ordering of the phases is essential to obtain correct results, within each phase the different ghost zones and patches may be processed in any order.

3. Use symmetry operations to fill in the corners of all symmetry ghost zones in all patches.

### 3. Jacobian Computation

The symbolic-differentiation Jacobian (14) must be modified to take into account the ghost-zone symmetry operations and interpatch interpolations described in the previous subsection. This is essentially a straightforward application of the chain rule for each of the  $D_u$  and  $D_{uv}$  terms in (14). Figure 6 shows the resulting algorithm in detail.

## APPENDIX B: MULTIPROCESSOR AND PARALLELIZATION ISSUES

## APPENDIX C: SEARCHING FOR THE CRITICAL PARAMETER OF A 1-PARAMETER INITIAL DATA SEQUENCE

In the interests of brevity these two appendices are omitted here. They appear as supplemental material in the online edition of this paper, and as appendices B and C in the preprint-archive version (gr-qc/0306056).

## ACKNOWLEDGEMENTS

I thank the Alexander von Humboldt foundation, the AEI visitors program, and the AEI postdoctoral fellowship program for financial support. I thank Peter Diener and Ian Hawke for useful conversations. I thank Peter Diener, Ian Hawke, Scott Hawley, Denis Pollney, Ed Seidel, and Erik Schnetter for helpful comments on various drafts of this paper. I thank two anonymous referees for a number of helpful comments.

I thank Frank Herrmann for providing the last sample binary-black-hole evolution discussed in section XC. I thank Tom Goodale, Thomas Radke, and many others for their assistance with the invaluable CACTUS computational toolkit. I thank Thomas Radke for our fruitful collaboration on CACTUS interpolators, and in particular for his work on the PUGHINTERP global interpolator. I thank Erik Schnetter for supplying the CARPET mesh-refinement driver and CARPETINTERP global interpolator for CACTUS. I thank P. Maderom and Tom Nicol for supplying the ILUCG sparse-matrix subroutine.

- 
- [1] M. Alcubierre, S. Brandt, B. Brügmann, C. Gundlach, J. Massó, E. Seidel, and P. Walker. Test-beds and applications for apparent horizon finders in numerical relativity. *Class. Quantum Grav.*, 17:2159–2190, 2000.
  - [2] M. Alcubierre and B. Brügmann. Simple excision of a black hole in 3+1 numerical relativity. *Phys. Rev. D*, 63:104006, 2001.
  - [3] Miguel Alcubierre, Bernd Brügmann, Denis Pollney, Edward Seidel, and Ryoji Takahashi. Black hole excision for dynamic black holes. *Phys. Rev. D*, 64:61501 (R), 2001.



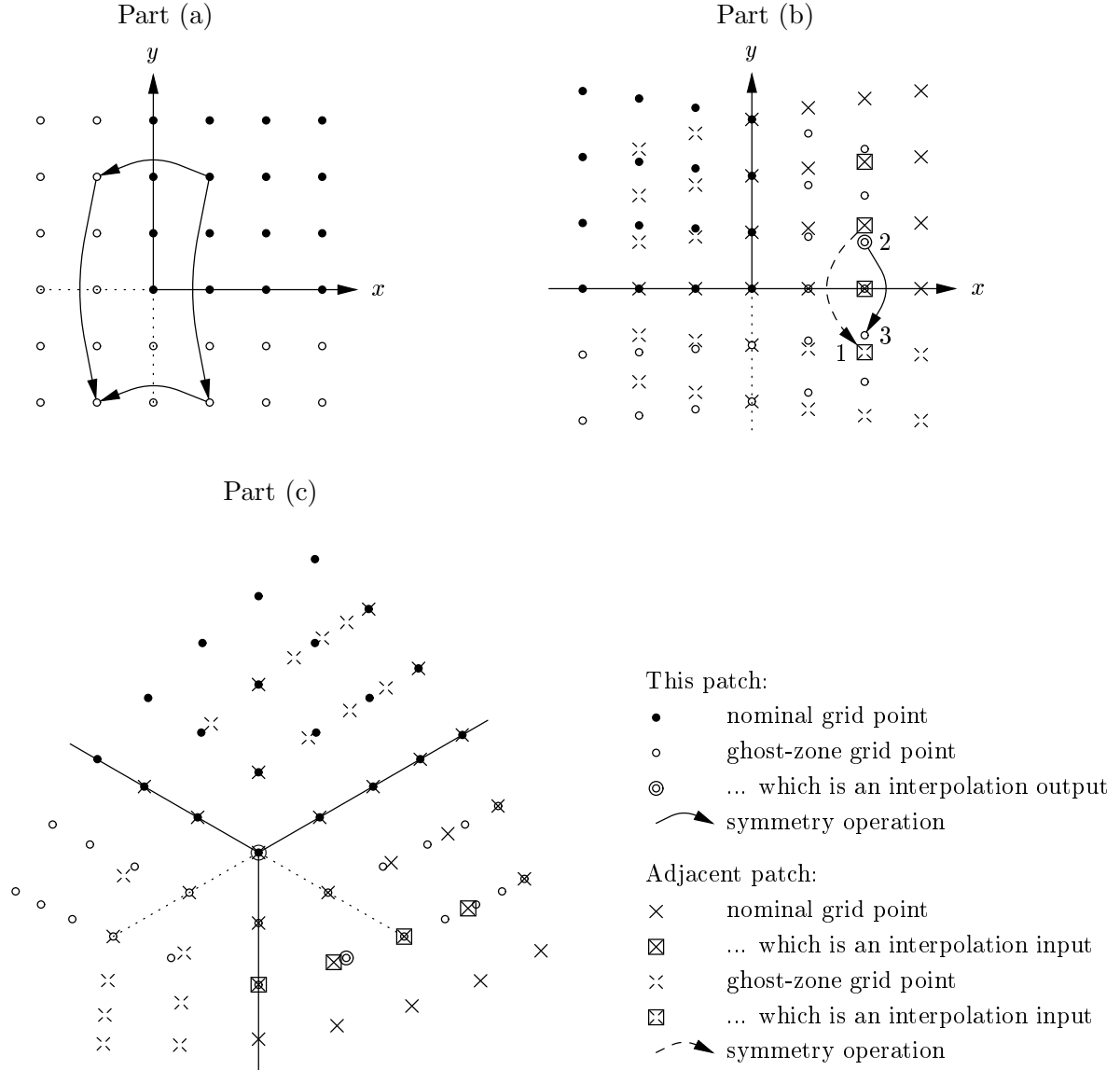


FIG. 5: This figure shows the combinations of symmetry operations and interpatch interpolations used to fill in grid function values in ghost-zone corners. Part (a) (where there are both  $x \leftrightarrow -x$  and  $y \leftrightarrow -y$  reflection symmetries) shows a corner between two symmetry ghost zones. Part (b) (where there is a  $y \leftrightarrow -y$  reflection symmetry) shows a corner between a symmetry and an interpatch ghost zone; sample output points for each phase of the 3-phase algorithm described in the text are labelled as 1, 2, and 3. Part (c) (where there are no symmetries) shows a corner between two interpatch ghost zones (this only happens when 3 patches meet at a corner). In each part, arrows show the symmetry operations, and for parts (b) and (c) the boxed and circled points show the inputs and outputs for the interpatch interpolations.

```

J ← 0 matrix
for each angular grid point I
{
    for each angular coordinate index u and pair of indices uv
    {
        for each molecule index m ∈ Du or Duv respectively
        {
            J ← I + m
            temp ←  $\frac{\partial \Theta}{\partial (\partial_u h)} D_u[\mathbf{m}]$  or  $\frac{\partial \Theta}{\partial (\partial_{uv} h)} D_{uv}[\mathbf{m}]$  respectively
            if (J ∈ nominal grid of the patch containing I)
                then JIJ ← JIJ + temp
            else {
                for each angular grid point K used in computing h[J]
                via the 3-phase algorithm of appendix A 2
                {
                    JIK ← JIK + temp ×  $\left( \frac{\partial h[\mathbf{J}]}{\partial h[\mathbf{K}]} \text{ for the 3-phase algorithm} \right)$ 
                }
            }
        }
    }
}
JII ← JII +  $\frac{\Theta(h + \varepsilon) - \Theta(h)}{\varepsilon}$ 
}

```

FIG. 6: This figure shows AHFINDERDIRECT’s overall Jacobian-computation algorithm, including ghost-zone handling.

- [4] P. Anninos, D. Bernstein, S. Brandt, J. Libson, J. Massó, E. Seidel, L. Smarr, W.-M. Suen, and P. Walker. Dynamics of apparent and event horizons. *Phys. Rev. Lett.*, 74:630–633, 1995.
- [5] P. Anninos, Karen Camarda, Joseph Libson, J. Massó E. Seidel, and W.-M. Suen. Finding apparent horizons in dynamic 3D numerical spacetimes. *Phys. Rev. D*, 58:24003, 1998.
- [6] P. Anninos, G. Daues, J. Massó, E. Seidel, and W.-M. Suen. Horizon boundary conditions for black hole spacetimes. *Phys. Rev. D*, 51(10):5562–5578, 1995.
- [7] A. Ashtekar, C. Beetle, and S. Fairhurst. Isolated horizons: A generalization of black hole mechanics. *Class. Quantum Grav.*, 16:L1–L7, 1999.
- [8] R. E. Bank and D. J. Rose. Parameter selection for Newton-like methods applicable to nonlinear partial differential equations. *SIAM Journal of Numerical Analysis*, 17(6):806–822, 1980.
- [9] R. E. Bank and D. J. Rose. Global approximate Newton methods. *Numerische Mathematik*, 37:279–295, 1985.
- [10] T. W. Baumgarte, G. B. Cook, M. A. Scheel, S. L. Shapiro, and S. A. Teukolsky. Implementing an apparent-horizon finder in three dimensions. *Phys. Rev. D*, 54(8):4849–4857, 1996.
- [11] S. Brandt, R. Correll, R. Gómez, Mijan F. Huq, P. Laguna, L. Lehner, P. Marronetti, R. A. Matzner, D. Neilsen, J. Pullin, E. Schnetter, D. Shoemaker, and J. Winicour. Grazing collisions

- of black holes via the excision of singularities. *Phys. Rev. Lett.*, 85:5496–5499, 2000.
- [12] D. Brill and R. Lindquist. Interaction energy in geometrostatics. *Phys. Rev.*, 131:471–476, 1963.
  - [13] Scott A. Caveny, Matthew Anderson, and Richard A. Matzner. Tracking black holes in numerical relativity. 2003. preprint gr-qc/0303099.
  - [14] M. W. Choptuik. Consistency of finite-difference solutions to Einstein’s equations. *Phys. Rev. D*, 44:3124, 1991.
  - [15] G. B. Cook, M. F. Huq, S. A. Klasky, M. A. Scheel, A. M. Abrahams, A. Anderson, P. Anninos, T. W. Baumgarte, N.T. Bishop, S. R. Brandt, J. C. Browne, K. Camarda, M. W. Choptuik, R. R. Correll, C. R. Evans, L. S. Finn, G. C. Fox, R. Gómez, T. Haupt, L. E. Kidder, P. Laguna, W. Landry, L. Lehner, J. Lenaghan, R. L. Marsa, J. Masso, R. A. Matzner, S. Mitra, P. Papadopoulos, M. Parashar, L. Rezzolla, M. E. Rupright, F. Saied, P. E. Saylor, E. Seidel, S. L. Shapiro, D. Shoemaker, L. Smarr, W. M. Suen, B. Szilagyi, S. A. Teukolsky, M. H. P. M. van Putten, P. Walker, J. Winicour, and J. W. York Jr. Boosted three-dimensional black-hole evolutions with singularity excision. *Phys. Rev. Lett.*, 80:2512–2516, 1998.
  - [16] A. R. Curtis and J. K. Reid. The choice of step lengths when using differences to approximate Jacobian matrices. *J. Inst. Math. Appl.*, 13(1):121–126, 1974.
  - [17] T. A. Davis. Algorithm 8xx: UMFPACK V3.2, an unsymmetric-pattern multifrontal method with a column pre-ordering strategy. Technical Report TR-02-002, Univ. of Florida, CISE Dept., Gainesville, FL, January 2002. ([www.cise.ufl.edu/tech-reports](http://www.cise.ufl.edu/tech-reports). Submitted to *ACM Trans. Math. Softw.*).
  - [18] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. Technical Report TR-02-001, Univ. of Florida, CISE Dept., Gainesville, FL, January 2002. ([www.cise.ufl.edu/tech-reports](http://www.cise.ufl.edu/tech-reports). Submitted to *ACM Trans. Math. Softw.*).
  - [19] Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Applic.*, 18(1):140–158, 1997.
  - [20] Timothy A. Davis and Iain S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, 25(1):1–19, 1999.
  - [21] P. Diener. A new general purpose event horizon finder for 3D numerical spacetimes. *Class. Quantum Grav.*, 20:4901–4917, 2003. gr-qc/0305039.
  - [22] Olaf Dreyer, Badri Krishnan, Deirdre Shoemaker, and Erik Schnetter. Introduction to isolated horizons in numerical relativity. *Physical Review D*, 67:024018, 2002.
  - [23] R. Gómez, L. Lehner, P. Papadopoulos, and J. Winicour. The eth formalism in numerical relativity. *Classical and Quantum Gravity*, 14:977–990, 1997.
  - [24] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing - VECPAR’2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.
  - [25] C. Gundlach. Pseudo-spectral apparent horizon finders: An efficient new algorithm. *Phys. Rev. D*, 57:863–875, 1998. gr-qc/9707050.
  - [26] S. W. Hawking and G. F. R. Ellis. *The Large Scale Structure of Spacetime*. Cambridge University Press, Cambridge, England, 1973.
  - [27] Mijan F. Huq. *Apparent Horizon Location in Numerical Spacetimes*. PhD thesis, The University of Texas at Austin, 1996.
  - [28] Mijan F. Huq, M. W. Choptuik, and Richard A. Matzner. Locating boosted Kerr and Schwarzschild apparent horizons. *Phys. Rev. D*, 66:084024, 2002. gr-qc/0002076.

- [29] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, 1978.
- [30] A. J. Kemball and N. T. Bishop. The numerical determination of apparent horizons. *Class. Quantum Grav.*, 8:1361–1367, 1991.
- [31] David S. Kershaw. The incomplete Cholesky – conjugate gradient method for iterative solution of linear equations. *J. Comp. Phys.*, 26(1):43–65, January 1978.
- [32] Joseph Libson, Joan Massó, Edward Seidel, Wai-Mo Suen, and Paul Walker. Event horizons in numerical relativity: Methods and tests. *Phys. Rev. D*, 53(8):4335–4350, 1996.
- [33] C. Misner. Wormhole initial conditions. *Phys. Rev. D*, 118(4):1110–1111, 1960.
- [34] C. W. Misner. The method of images in geometrostatics. *Ann. Phys.*, 24:102, 1963.
- [35] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. W. H. Freeman, San Francisco, 1973.
- [36] Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratory, Argonne, USA, August 1980. available from the NETLIB online software repository, <http://www.netlib.org/minpack/>.
- [37] T. Nakamura, Y. Kojima, and K. Oohara. A method of determining apparent horizons in three-dimensional numerical relativity. *Phys. Lett.*, 106A:235–238, 1984.
- [38] U. Nowak and L. Weimann. GIANT – a software package for the numerical solution of very large systems of highly nonlinear equations. Technical Report TR-90-11, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), December 1990.
- [39] U. Nowak and L. Weimann. A family of Newton codes for systems of highly nonlinear equations. Technical Report TR-91-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), December 1991.
- [40] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes*. Cambridge University Press, New York, 2nd edition, 1992.
- [41] Erik Schnetter. A fast apparent horizon algorithm. 2002. gr-qc/0206003.
- [42] Erik Schnetter. Finding apparent horizons and other two-surfaces of constant expansion. 2003. gr-qc/0306006.
- [43] M. R. Schroeder. *Number Theory in Science and Communication*. Springer-Verlag, Berlin, 1986.
- [44] E. Seidel and W.-M. Suen. Towards a singularity-proof scheme in numerical relativity. *Phys. Rev. Lett.*, 69(13):1845–1848, 1992.
- [45] Deirdre M. Shoemaker, Mijan F. Huq, and Richard A. Matzner. Generic tracking of multiple apparent horizons with level flow. *Phys. Rev. D*, 62:124005 (12 pages), 2000.
- [46] Ulrich Sperhake, Kenneth L. Smith, Bernard Kelly, Pablo Laguna, and Deirdre Shoemaker. Impact of densitized lapse slicings on evolutions of a wobbling black hole. July 2003. gr-qc/0307015.
- [47] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, Berlin and New York, 1980.
- [48] Jonathan Thornburg. Finding apparent horizons in numerical relativity. *Phys. Rev. D*, 54(8):4899–4918, 1996.
- [49] Jonathan Thornburg. A 3+1 computational scheme for dynamic spherically symmetric black hole spacetimes – I: Initial data. *Phys. Rev. D*, 59, 1999.
- [50] Jonathan Thornburg. A multiple-grid-patch evolution scheme for 3-D black hole excision. In V. Gurzadyan, R. T. Jantzen, and R. Ruffini, editors, *The Ninth Marcel Grossman Meeting: On Recent Developments in Theoretical and Experimental General Relativity, Gravitation,*

- and Relativistic Field Theories*. World Scientific, Singapore, 2003. gr-qc/0012012.
- [51] K. P. Tod. Looking for marginally trapped surfaces. *Class. Quantum Grav.*, 8:L115–L118, 1991.
  - [52] A. Čadež. Apparent horizons in the two-black-hole problem. *Ann. Phys.*, 83:449–457, 1974.
  - [53] R. M. Wald. *General Relativity*. The University of Chicago Press, Chicago, 1984.
  - [54] J. York. Kinematics and dynamics of general relativity. In L. Smarr, editor, *Sources of Gravitational Radiation*. Cambridge University Press, Cambridge, England, 1979.
  - [55] J. York. Initial data for collisions of black holes and other gravitational miscellany. In C. Evans, L. Finn, and D. Hobill, editors, *Frontiers in Numerical Relativity*, pages 89–109. Cambridge University Press, Cambridge, England, 1989.